EF255373033US

APPLICATION DATA SHEET

FOR

UNITED STATES LETTERS PATENT

APPLICANTS: Robert J. Devins
Paul G. Ferro
Emory D. Keller
David W. Milton

FOR: SYSTEM FOR CONTROLLING EXTERNAL MODELS USED FOR VERIFICATION OF SYSTEM ON A CHIP (SOC) INTERFACES

DOCKET NO.: BUR9-2001-0016-US1

INTERNATIONAL BUSINESS MACHINES CORPORATION

SYSTEM FOR CONTROLLING EXTERNAL MODELS USED FOR VERIFICATION OF SYSTEM ON A CHIP (SOC) INTERFACES

BACKGROUND OF THE INVENTION

Field of the Invention

[0001] The present invention relates generally to the verification of integrated circuit (IC) logic, and more particularly to a method and system for increasing the efficiency and reusability of verification software and the verification environment.

Description of the Related Art

[0002] Before ICs are released to market, the logic designs incorporated therein are typically subject to a testing and de-bugging process known as "verification." Verification of logic designs using simulation software allows a significant number of design flaws to be detected and corrected before incurring the time and expense needed to physically fabricate designs.

[0003] Hardware verification typically entails the use of software "models" of design logic. Such models may be implemented as a set of instructions in a hardware description language (HDL). The models execute in a simulation environment and can be programmed to simulate a corresponding hardware implementation. The simulation environment comprises

specialized software for interpreting model code and simulating the corresponding hardware device or devices. By applying test stimuli (typically in batches known as "test cases") to a model in simulation, observing the responses of the model and comparing them to expected results, design flaws can be detected and corrected.

[0004] Advances in technology have permitted logic designs to be packed with increased density into smaller areas of silicon as compared with past IC devices. This has led to "system-on-a-chip" (SOC) designs. The term "SOC" as used herein refers to combinations of discrete logic blocks, often referred to as "cores," each performing a different function or group of functions. A SOC integrates a plurality of cores into a single silicon device, thereby providing a wide range of functions in a highly compact form. Typically, an SOC is comprised of a processor core (often referred to as an "embedded" processor), and will further comprise one or more cores for performing a range of functions often analogous to those of devices in larger-scale systems.

[0005] In its developmental stages, a core is typically embodied as a simulatable HDL model written at some level of abstraction, or in a mixture of abstraction levels. Levels of abstraction that are generally recognized include a behavioral level, a structural level, and a logic gate level. A core may be in the form of a netlist including structural and logic gate elements or a behavioral model.

[0006] Verification of a SOC presents challenges because of the number of cores and the complexity of interactions involved, both between the cores internally to the SOC, and between the SOC and external logic. An acceptable level of verification demands that a great number of test cases be applied, both to individual components of the SOC, and to the cores interconnected as a system and interfacing with logic external to the SOC. There is a commensurate demand on

computer resources and time. Accordingly, techniques which increase the efficiency of verification are at a premium.

[0007] According to one standard technique, already-verified models are used to test other models. The electronic design automation (EDA) industry has reached a level of sophistication wherein vendors offer standardized models for use in verification of other models still in development. In particular, such models are typically used for testing cores in a SOC that have external interfaces (i.e., communicate with logic external to the SOC). Such standardized models save the purchaser development resources, are typically well-tested and reliable, and are designed to have a wide range of applicability.

[0008] However, there are disadvantages associated with using standardized models. For instance, they can be very costly. Moreover, they can be very complex and provide much more functionality than is needed by the purchaser if only a subset of functions are required. Further, the standardized models must be integrated into existing verification systems, incurring more cost in terms of time and effort.

[0009] Alternatively to purchasing and using standardized models, designers may, of course, develop their own testing models. However, this is costly in terms of development time, with the typical result that such models are designed for limited application. Accordingly, they typically have limited functionality and reusability.

SUMMARY OF THE INVENTION

[0010] In view of the foregoing and other problems, disadvantages, and drawbacks of the conventional verification test benches, the present invention has been devised, and it is an object of the present invention to provide a structure that attaches an external model to a SOC interface and to an external bus interface unit.

[0011] In order to attain the object suggested above, there is provided, according to one aspect of the invention, a verification test bench system. This system tests an interface of a system-on-a-chip (SOC). The verification test bench system comprises a verification interface model which is connected to the SOC interface and a test bench external bus interface unit (EBIU) connected to the interface model. The test bench EBIU is connected to the SOC EBIU. The SOC interface and verification interface models are programmable by a test case that runs in the SOC. The test bench EBIU allows a test case to control both the SOC interface and the interface model. The test case can use either the same or different software drivers to configure and control the SOC interface and verification interface models.

[0012] The invention represents a clean way of controlling external interface models without the need for a complex control mechanism such as the conventional semaphore derived scheme used to enable communication between the SOC being tested and the external interface model. The external bus mastering of the test bench EBIU 200 allows external model programming from the SOC test case. Thus, the same test case directly controls operations of the SOC interface 101 and the external model 210.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] The foregoing and other objects, aspects and advantages will be better understood from the following detailed description of the preferred embodiments of the invention with reference to the drawings, in which:

[0014] Figure 1 is a schematic diagram illustrating connections between an SOC and an external verification model comprising an interface model and an external bus interface unit; and

[0015] Figure 2 is a schematic diagram of a computer system which can be used to implement the present invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS OF THE INVENTION

[0016] The invention provides a structure that controls an external model(s) used for interconnection verification of an SOC interface. An important feature of the invention is to connect the external model to an external bus interface unit (EBIU) via a local bus. The EBIU is then connected to the SOC's EBIU. This connection enables the external model to be completely controlled from the test case running in the SOC by using the EBIU's external bus mastering capability. The EBIU in this invention is not a specific core or unit and is meant to refer to any communications channel that can connect the SOC to the external model and provide the external bus mastering functionality. This allows the test case to program the external model to perform

such tasks as data transfers and other interface specific functions necessary to thoroughly verify the SOC interface. The same test case is also used to program the SOC interface.

[0017] Present methods for controlling external models generally involve decoupled control between the test case and the external model(s). They are essentially different entities within the test bench written in separate control languages (HDL or Bus Functional Models) that require their own separate programming languages, and thus require different software to control each.

[0018] Additionally, test coordination between the test case and the external model involve complex communication schemes such as semaphore types where the test case running on the SOC will set a flag or write a value to a test bench memory model or register array that the external model can read as a signal to perform some operation. Alternatively, the external model simply runs through its own "pre-canned" operations such as sending test data to the SOC interface when it is turned on by the start of a test in the test bench.

[0019] To the contrary, the present invention implements control by attaching an EBIU 200 to an external interface model 210 (e.g., a verification interface model) via a local bus structure 201, as shown in Figure 1. This essentially creates enough of an external SOC structure 300 (e.g., a verification test bench) that is capable of having data or commands transferred from an external source (the SOC 100 in this case) through the test bench EBIU 200 into or through the verification interface model 210.

[0020] The test bench EBIU 200 attaches to the SOC EBIU 205 via an external bus 206. The internal bus structure 201 within the verification test bench 300 is implemented to respond to an address range that is unique to the SOC, giving complete control to a single verification test

running in the SOC 100. In addition, Figure 1 illustrates various internal components of the SOC 100 including the central processing unit (CPU) 130 and the test patterns 135-137.

[0021] Item 215 represents an extra external interface model that the invention can be optionally used to test more than one type of SOC interface. Items 135-137 represent different software drivers (SWD) for driving or configuring different interfaces of the SOC. A software driver is software written only for a specific hardware device like a printer or a specific interface of an SOC. The test cases are written to utilize software drivers to configure the core or units they are testing.

[0022] In operation, the structure in Figure 1 utilizes the interface model 210 to test the SOC's interface 101. A bus master can configure other units or cores connected to a bus, whereas a bus slave can only respond to a master's commands. The EBIU 200 (slave) responds to external bus master commands from the CPU 130, thereby enabling a test case running in the SOC 100 to direct the CPU 130 (which is a master on the SOC's internal bus 131) to act as a master on the test bench's internal bus 201. Thus, the test case running in the SOC 100 can direct the SOC CPU 130 to program (e.g., master) the registers in both the internal interface 101 and the interface model 210. Here the same test case software is used to program both the SOC's interface 101 and the external entity's 300 interface model 210.

[0023] In Figure 1, the invention transfers data from the external interface model 210 to the SOC interface 101. The test case calls the software driver (SWD) 135-137 for the interface 101 and instructs the software driver to configure the interface 101 to receive data. Next, the test case calls the same SWD 135-137 and instructs the software driver to configure the external interface model 210 to send data. The SOC's interface 101 and the external interface model 210

are implemented to respond to different unique addresses. Thus, when the test case calls the SWD 135-137 to perform some configuration on one of the interfaces, the test case sends the address of that interface along with the operation to be performed. The test case then sends test data to the unique data address of the external interface model 210. This data is sent from the SOC 100 through the SOC's external bus interface unit (EBIU) 205 to the external EBIU 200 and then along to the interface model 210. From there the data is sent through the interface model 210 into the SOC's interface 101 which is configured to receive data. Once the data is back in the SOC 100, the test case checks it for correctness and a test status is recorded.

[0024] One advantage achieved with the invention is better software control. The invention allows the test case executing in the SOC 100 to use the same software driver, if appropriate, to program both interfaces 101, 210. The test case can also use one driver to program the SOC interface 101 and another to program the interface model 210, both of which are controlled by the test case. This is an improvement over the conventional situation where the test case running within the SOC 100 controls the SOC interface 101, and another software program (written in a bus functional language) controls the external interface 210. The invention provides increased reusability and decreased development time because the invention uses the same or similar software written in the same language to program both interfaces.

[0025] Figure 2 shows a computer system which can be used to implement the present invention. The system includes a computer 400 comprising a memory 401 and processor 402 which may be embodied, for example, in a workstation. The system may include a user interface 403 comprising a display device 404 and user input devices such as a keyboard 405 and mouse 406. The verification test bench 450 may be implemented as computer-executable instructions

which may be stored on computer-usable media such as disk storage 407, CD-ROM 408, magnetic tape 409 or diskette 410. The instructions may be read from a computer-usable medium as noted into the memory 401 and executed by the processor 402 to effect the advantageous features of the invention. A simulator 411 loaded into computer memory 401 and executed by processor 402 interprets a compiled verification test bench to simulate hardware devices corresponding thereto. The simulator 411 may be any of a variety of commercially available simulators, including event simulators, cycle simulators and instruction set simulators. Programming structures and functionality implemented in computer-executable instructions as disclosed herein-above for performing steps of the method may find specific implementations in a variety of forms, which are considered to be within the abilities of a programmer of ordinary skill in the art.

[0026] The invention represents a clean way of controlling external interfaces without the need for a complex control mechanism such as the conventional semaphore derived scheme used to enable communication between the SOC being tested and the external interface. The external bus mastering of the test bench EBIU 200 allows external model programming from the SOC test case. Thus, the same test case directly controls operations of the SOC interface 101 and the external model 210. Sometimes the external model needed to verify the SOC interface 101 comprises a different type of interface such that it cannot be programmed with the same driver. The invention readily accommodates such interfaces by also having the ability to individually program the non-conforming external interface using a separate, appropriate driver. Thus, the invention achieves an advantage by effectively controlling external models 210 that are necessary to verify the interconnection of SOC external interfaces 101 in an efficient and simple manner.

[0027] While the invention has been described in terms of preferred embodiments, those skilled in the art will recognize that the invention can be practiced with modification within the spirit and scope of the appended claims.